Optimized Trajectories of Multi-Robot Deploying Wireless Sensor Nodes

Ines Khoufi*, Mohamed Hadded†, Pascale Minet* and Anis Laouiti†
* Inria, 78153 Le Chesnay Cedex, France
† TELECOM SudParis, CNRS Samovar UMR 5157 91011 Evry Cedex, France
ines.khoufi@inria.fr, mohamed.hadded@telecom-sudparis.eu, pascale.minet@inria.fr, anis.laouiti@telecom-sudparis.eu

***Abstract*—A main reason to the growth of wireless sensor networks deployed worldwide is their easy and fast deployment. In this paper we consider deployments assisted by mobile robots where static sensor nodes are deployed by mobile robots in a given area. Each robot must make a tour to place its sensor nodes. All sensor nodes must be placed at their precomputed positions. The Multi-Robot Deploying wireless Sensor nodes problem, called the MRDS problem, consists in minimizing the longest tour duration (i.e. the total deployment duration), the number of robots used and the standard deviation between duration of robots tours. After a formal definition of the MRDS problem, we show how to use a multi-objective version of genetic algorithms, more precisely the NSGA-II algorithm, to solve this multi-objective optimization problem. The solutions belonging to the best Pareto front are given to the designer in charge of selecting the best trade-off taking into account various criteria. We then show how to extend this method to take obstacles into account, which is more representative of real situations.**

I. Introduction and motivation

Wireless Sensor Networks (WSNs) are widely used in monitoring applications due to their efficiency in detecting and reporting events. Area coverage and network connectivity are two major issues in the monitoring task. An area is considered covered if each event occurring in the area can be detected by at least one sensor node. Network connectivity is maintained if the detected event can be reported to the sink. Coverage and connectivity are related to sensor node positions in the area considered. For instance, if sensor nodes are densely deployed, multiple coverage (i.e. a zone is covered by several sensors) can be ensured and many paths from each node to the sink can be maintained. In a sparse deployment, coverage holes may exist and network connectivity may fail. Then, a deployment algorithm that determines the appropriate sensor node positions is needed to meet the coverage and connectivity requirements. We distinguish between two types of deployment: autonomous self-deployment and assisted deployment. In a self-deployment, sensor nodes are mobile and autonomous, they cooperate to determine their final positions. However, if the deployment is assisted, then sensor node positions are precomputed and a human or a mobile robot is in charge of placing static sensor nodes at their positions. When designing a deployment, some constraints should be taken into account. First, the number of nodes deployed should be minimized. Second, if the deployment is assisted, the time needed to place sensor nodes should also be minimized, mainly if the environment is hostile or if the robot is battery operated. Third, obstacles are always present in the area considered. They have an impact on the robot trajectory when deploying sensor nodes or on network connectivity: if the obstacle is opaque, the communication between two nodes at a distance less than the communication range may fail.

In this paper, we focus on a deployment assisted by mobile robots that copes with the presence of obstacles and optimizes the deployment duration by minimizing the longest robot tour duration, minimizes the number of robots used and balances the robot tour duration. We then show how to optimize the tours of robots, while avoiding obstacles in the area considered, which corresponds to more realistic configurations.

II. State of the art

The reason to use one or several robots to deploy sensor nodes is that a high number of autonomous and mobile sensor nodes may be too expensive. In the assisted deployment, we distinguish between two different situations where mobile robots are in charge of deploying static sensor nodes.

In the first situation: the robot has two tasks: it should on the one hand move and discover the area considered and on the other hand place sensor nodes at their positions to meet the coverage and connectivity requirements. A robot has to follow predefined rules to move in the area considered and place sensor nodes. This strategy is proposed in [1] where one robot follows a spiral movement policy to deploy static sensor nodes along its trajectory. The goal is not to optimize the robot trajectory but to ensure full area coverage and network connectivity using the minimum number of sensor nodes. In addition, some movement policies are defined to permit the robot to bypass the obstacles. In a similar context, authors in [2], propose a serpentine movement policy with obstacle handling policy and boundary policy. The robot has to follow the serpentine movement policy while placing static sensor nodes separated by the optimal distance to reduce the total number of sensor nodes. To conclude, in such a situation,

the policies proposed in the two papers cited permit the robot to visit the whole area while avoiding obstacles and placing sensor nodes.

In the second situation: sensor node positions are precomputed and given to the robot(s). In this situation, each position should be visited by exactly one robot and a sensor node should be placed at each position computed. Then, the problem is different. The goal is no longer to discover the area considered and compute node positions, but the problem is how to optimize the duration needed to deploy these sensor nodes. When one robot is involved, this problem becomes similar to the Traveling Salesman Problem, TSP [3], whose goal is to find the shortest tour visiting all sensor nodes positions (representing the cities) only once and going back to its initial position. In [4], we propose the RDS problem, Robot Deploying Sensor, where one robot is in charge of placing sensor nodes at their precomputed positions. The RDS problem is based on the principle of TSP. However, the RDS problem aims to optimize robot tour duration instead of minimizing the total distance traveled. The robot tour duration includes not only the time needed to travel the distance between node positions but also the time needed to change the direction. Since the deployment duration should be minimized in order to save robot battery and in case of hostile environment to prolong its lifetime, more than one robot may be needed. A game theory approach is used in [5] to find the optimized tours in terms of deployment duration of two robots in charge of deploying static sensor nodes. In this study, the robots have not only to place sensor nodes at their appropriate positions but also to avoid opaque obstacles. The proposed approach meet some constraints such as the robots can carry a limited number of sensor nodes.

The Vehicle Routing Problem (VRP) [6] generalizes the Traveling Salesman Problem. The vehicle routing problem aims to find a set of tours that visits all positions at a minimal cost such as the shortest path, the minimum number of vehicles, etc. The vehicles start and end their tours at the depot. Each position is visited only once, by only one vehicle, and each vehicle has a limited capacity.

Our problem, called the Multi-Robot Deploying wireless Sensor nodes (MRDS) problem, presents many similarities with the VRP problem: mobile robots correspond to vehicles and Points of Interest (PoIs) where sensor nodes should be placed to ensure the monitoring task, correspond to the customers to be served. However, there are differences on the objectives to optimize as we will see in the next section.

III. DEPLOYMENT ASSISTED BY ROBOTS

A. Goal

Our goal is to minimize the deployment duration of static sensor nodes, called Points of Interest, in a given environment by $K \geq 1$ mobile robots. Since on the one hand robots are battery-operated and on the other hand the environment may be hostile (e.g. deployment in a post-crisis situation), the duration of the deployment must be the shortest as possible. Hence the idea of minimizing the duration of the longest tour performed by a robot. In addition, the best balancing between robot tours durations is required. Sensor node positions are computed such that area coverage and network connectivity are ensured. Then, there is at least one path from each sensor node to the sink in order to forward the collected data.

B. Problem formalization

The Multi-Robot Deploying wireless Sensor nodes (MRDS) problem is defined as follows:

Let $\{1, \dots N\}$ be the set of PoIs to be visited by robots. By convention, 0 is called the depot. It is the departure and arrival location of robots. Let $K \geq 1$ be the number of available robots. The problem is to design a set of k tours, one tour per robot with $1 \leq k \leq K$, that:

- Minimizes the longest tour duration,
- Minimizes the number of robots used,
- Minimizes the standard deviation of the robot tour durations.

Under the constraints:

- Any robot k, with $1 \leq k \leq K$, has a limited carriage capacity Q_k: it is unable to carry more than Q_k sensors.
- Each robot starts and ends its tour at the depot.
- Each PoI should be visited by exactly one robot.

The MRDS problem can be expressed more formally with the following notation:
N: is the total number of PoIs to be visited, $K \geq 1$ is the number of available robots and K^* is the number of robots really used. Thus, we have $1 \leq K^* \leq K$. The depot is denoted by 0, and the PoIs are denoted by 1, 2 or N.
Q_k: is the capacity of robot k.
$d_{i,j}$: is the distance required for traveling from node i to node j.
l_s: is the linear speed of each robot.
a_s: is the angular speed of each robot.
$\hat{\theta}_{i,j,t}$: is the angle formed by the segments $[i, j]$ and $[j, t]$.

The decision variables of the model are:
X_{ij}^k: is the decision variable that is equal to 1 if robot k visits PoI j immediately after PoI i and is equal to 0 otherwise.
Y_i^k: is the decision variable that is equal to 1 if PoI i is visited by robot k and is equal to 0 otherwise.

Let TT_k be the tour duration of the robot k. This duration includes the duration due to the distance traveled and the duration due to direction changes.

$$TT_k = \sum_{i=0}^{N}\sum_{j=0}^{N} d_{i,j} * X_{ij}^k/ls + \sum_{i=0}^{N}\sum_{j=1}^{N}\sum_{t=0}^{N} \hat{\theta}_{i,j,t} * X_{ij}^k * X_{jt}^k/a_s \quad (1)$$

First objective: minimizing the longest tour TT:

$$Minimize\left(TT = \max_{k\in[1,K]} TT_k\right) \quad (2)$$

Second objective: minimizing the number of robots used NT (i.e. number of tours):

$$Minimize\,(NT = K^*) \quad (3)$$

Third objective: minimizing the standard deviation σ of the robot tour durations:

$$Minimize(\sigma = \sqrt{\frac{1}{K^*}\left(\sum_{k=1}^{K^*} TT_k^2\right) - \left(\frac{1}{K^*}\sum_{k=1}^{K^*} TT_k\right)^2}) \quad (4)$$

Constraints:

- Each PoI is visited by exactly one robot

$$\sum_{k=1}^{K} Y_i^k = 1 \quad \forall i \in [1, N] \quad (5)$$

- The number of robots used is equal to $K^* \leq K$

$$\sum_{k=1}^{K}\sum_{i=1}^{N} Y_i^k = \sum_{k=1}^{K^*}\sum_{i=1}^{N} Y_i^k = N \quad (6)$$

- Each robot visits a number of PoIs less than its capacity

$$\sum_{i=1}^{N} Y_i^k \leq Q_k \quad \forall k \in [1, K^*] \quad (7)$$

- Subtours are eliminated

$$\sum_{i=0}^{N}\sum_{j=0}^{N} X_{ij}^k \leq \sum_{i=1}^{N} Y_i^k \quad \forall k \in [1, K^*] \quad (8)$$

- Decision variables $\in \{0, 1\}$

$$X_{ij}^k \in \{0,1\}, Y_i^k \in \{0,1\}, \forall i \in [1, N]; \forall k \in [1, K] \quad (9)$$

Thus, from equations 2, 3 and 4, the new MRDS problem is defined as follows,

$$Minimize\left(f_{\{TT,NT,\sigma\}} = \max_{k\in K}(TT_k), K^*, \sqrt{\frac{1}{K^*}\left(\sum_{k=1}^{K^*} TT_k^2\right) - \left(\frac{1}{K^*}\sum_{k=1}^{K^*} TT_k\right)^2}\right) \quad (10)$$

under the constraints 5 to 9 described above.

Property 1: A necessary feasibility condition of the MRDS problem is given by:

$$\sum_{k=1}^{K} Q_k \geq N. \quad (11)$$

IV. NSGA-II BASED APPROACH FOR MRDS OPTIMIZATION

A. Overview of NSGA-II

Multi-objective optimization (also known as multi-objective programming, vector optimization, multi-criteria optimization) is an area of multiple criteria decision making, that deals with mathematical optimization problems involving more than one objective function to be optimized simultaneously. Optimizing a group of objective functions is not a simple task. The Multi-objective Optimization Problem (MOP) can be formulated as follows:

$$(MOP)\begin{cases} min \quad f_i(x), \quad i \in [1, m] \\ \qquad\quad s.t \\ x \in D \end{cases}$$

Where the vector $x = (x_1, \ldots, x_n)^T \in D$ is the vector of n decision variables and m is the number of objectives. D is the feasible solution space, and $f_i(x)$ is an objective function, and the vector $y = (y_1, y_2, \ldots, y_m)$ is a solution, with $y_i = f_i(x)$.

Definition 1: For any MOP minimization, a solution $x \in D$ is said to be dominated by a solution $x' \in D$ (it is denoted by $x \prec x'$) if the following conditions are satisfied:

$i) f_i(x) \leq f_i(x')\ \forall\ i \in [1, m]$

$ii) \exists\ i \in [1, m]\ such\ \ that\ f_i(x) < f_i(x')$

The set of optimal solutions is composed of the non-dominated vectors, often called the Pareto front and also denoted $PF^* = \{x \in D \mid \exists\ x^{'} \in D, x^{'} \prec x\}$.
In other words, the Pareto front provides the best trade-off for the objectives considered. The goal of the multi-objective optimization is to find the Pareto front for a given problem. NSGA-II [7], Non dominated Sorting Genetic Algorithm, is often used to solve multi-objective optimization problems (see for instance, the use of NSGA-II to build clusters in vehicular networks [8]). This algorithm is a multi-objective version of the genetic algorithm in which the solutions explored are classified into Pareto-optimal fronts.

B. NSGA-II algorithm for the MRDS problem

1) Presentation of the NSGA-II algorithm: NSGA-II begins with an initial population *P* made up of solution vectors called individuals. At each iteration, an auxiliary population *Q* is formed by applying the crossover and mutation operators (lines 8 to 15). Then, both the current *P* and the new *Q* populations are merged together to form one set of solutions *R*, which will be sorted according to the non-domination and crowded comparison (line 17). Finally, only the best individuals in *R* are included in the next generation and will participate in the production step while the other individuals are deleted (lines 19 to 25). These steps are repeated until the maximum number of iterations is reached.

Algorithm 1 NSGA-II algorithm for MRDS problem

Input N *// population size*
P_c *//crossover probability*
P_m *// mutation probability*
$Nbr\_iteration\_max$ *// maximum number of iterations*

1: $Itr \leftarrow 0$ *// current iteration*
2: $P_{Itr} \leftarrow \{\emptyset\}$ *// population of iteration Itr*
3: initialize $P_{Itr=0} = \{\overrightarrow{x}^i_{Itr=0}, \ldots, \overrightarrow{x}^N_{Iter=0}\}$
4: evaluate $P_{Itr=0}$
5: **while** $(Itr < Nbr\_iteration\_max)$ **do**
6: $Q_{Itr} \leftarrow \{\emptyset\}$ *// new population*
7: $t \leftarrow 0$
8: **while** $(t \leq size(Q_{Itr})/2)$ **do**
9: $parents \leftarrow$ **selection**(P_{Itr})
10: $Child \leftarrow$ **crossover**$(P_c, parents)$
11: $E \leftarrow$ **mutation**$(P_m, Child)$
12: **compute_objective_values**$(Child)$
13: $Q_{Itr} \leftarrow Q_{Itr} \cup \{Child\}$
14: $t \leftarrow t + 1$
15: **end while**
16: $R_{Itr} \leftarrow P_{Itr} \cup \{Q_{Itr}\}$
17: $R_{Itr} = \bigcup_{i=1}^{r} F_i$ *//F_i is a Pareto front meeting* $F_1 < F_2 < \ldots < F_r$
18: $P_{Itr+1} \leftarrow \{\emptyset\}; i \leftarrow 0$
19: **while** $(|P_{Itr+1}| + |F_i| < N)$ **do**
20: $P_{Itr+1} \leftarrow P_{Itr+1} \cup F_i$
21: $i \leftarrow i + 1$
22: **end while**
23: **ranking**$(F_i, crowding\_distance)$
24: $Itr \leftarrow Itr + 1$
25: $P_{Itr} \leftarrow P_{Itr} \cup \{N - |P_{Itr}|\ first\ solutions\ in\ F_i\}$
26: **end while**

2) Application of NSGA-II to the MRDS problem:

Individual Representation: In MRDS problem, an individual represents a possible solution: a set of robots tours where the k^{th} robot makes the k^{th} tour, each tour is defined by the sequence of PoIs visited by a same robot. That is why we use two parts to define an individual. The first part of the individual, called sensor-part, is a set of integers where each integer represents a PoI visited by a robot. The second part of the individual, called the robot-part, contains robots information. The robot-part contains as many genes as tours (i.e. robots used) described in the sensor-part. In the robot-part, the k^{th} gene is equal to the number of PoIs visited by the k^{th} robot. An example of individual is depicted in Figure 1, where 3 robots are needed, robot 1 visiting 3 PoIs (4, 6 and 5 in this order). Robot 2 visiting the PoIs 1 and 2 and robot 3 the PoIs 3, 7 and 8 in this order.

Definition 2: An individual is said valid (i.e. it corresponds to a feasible solution) if and only if each PoI occurs exactly once in the sensor-part (in other words, the sensor part is a permutation of the sequence $1, 2, \ldots N$), the sum of the numbers included in the robot-part is equal to N the number of PoIs to visit and the size of the robot-part (i.e. number of robots really used) is less than or equal to K.

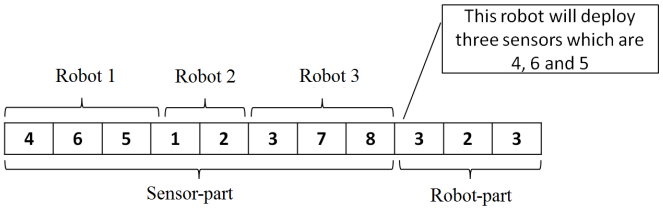


Fig. 1: An example of individual

Crossover: The crossover operator is one of the main parts of NSGA-II. The input of this operator consists of two solution vectors (known as parents). The output is two child vectors, which have certain features from both parents. For the sensor-part, we will apply the same crossover operator as used in the RDS problem [4], that is PMX (Partially Matched Crossover) due to its good performances [9].

Mutation: After recombination, the mutation operator is applied to randomly change some genes in an individual. This operator serves as a strategy to prevent solutions from being trapped in local optima. An example of mutation is illustrated in Figure 2, where the 8^{th} gene of the sensor-part of the individual described in Figure 1 suffers a mutation. This gene corresponding to PoI 8 belongs to the third robot (see Figure 2(1)). After mutation, this gene belongs to the third robot (i.e. PoI 8 is visited by robot 2), see Figure 2(2). To make the mutated individual valid, the robot-part of the individual is updated (minus 1 in the number of PoIs visited by robot 3 and plus 1 for robot 2), see Figure 2(3) depicting the new individual obtained after mutation of gene 8.

Selection: The selection process in NSGA-II is based on the dominance principle. The combined population P_{itr} is sorted Into Non-dominated levels $(F_1, .., F_r)$ according to the non-dominance principle. Solutions from the non-dominated front level 1 to fronts $l \leq r$ are included into the new population P_{itr+1}, such that $|P_{Itr+1}|$=population size. If some solutions of the last front F_l should be added to P_{itr+1}, these solutions are chosen such as they have the best crowding distance. The next generation is started based on the new population P_{itr+1}.

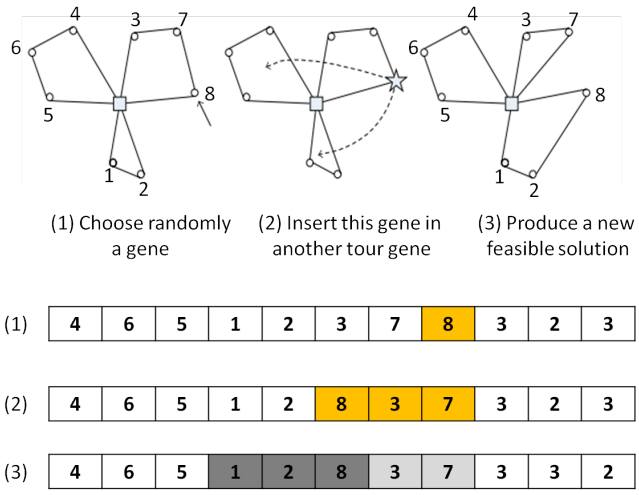


Fig. 2: An example of mutation operator

Selection of the best solutions: After the experimentation, we identify a set of Pareto optimal solutions by gathering all the non-dominated solutions found in 30 independent runs. These solutions provide various tradeoffs between the three objective functions considered. We let the designer choose the best tradeoff. For instance, if a solution with two robots is preferred, the designer will select the solution working with two robots providing the smallest deployment duration. If several solutions provide close tour durations, the designer may take the solution with the smallest standard deviation.

C. Hybrid algorithm combining NSGA-II and 2-opt for the MRDS problem

Figure 3 depicts a solution found by NSGA-II that minimizes the standard deviation for 3 robots and 20 PoIs. It is obvious that the duration of each tour in this figure is not minimized. There exists a permutation of the ordered sequence of PoIs visited that improves the tour duration.

More generally, NSGA-II needs many iterations to find good solutions, specially when it starts with initial solutions randomly chosen. We decided to provide NSGA-II with initial solutions that have already been optimized by a heuristic, in order to improve the quality of the solutions obtained by NSGA-II. We also noticed that for any given number of robots, any permutation of the sequence of PoIs visited that decreases the tour duration tends to improve the first objective (i.e. minimizing the longest tour duration), has no effect on the second objective (number of robots used), and has a positive impact on the third objective (the standard deviation of the robots tour durations). In other words, the new solution obtained dominates the initial one. That is why we choose the 2-Opt heuristic [10] to improve the tour duration of any individual given to or created by NSGA-II. The 2-Opt algorithm [10] starts with an initial

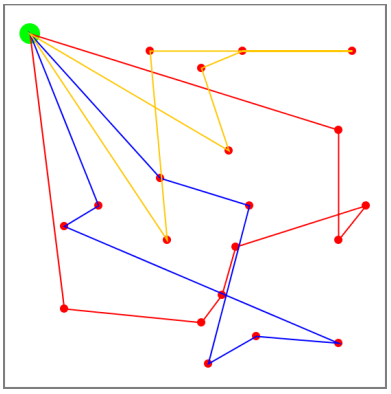


Fig. 3: Best solution minimizing the standard deviation for 3 robots and 20 PoIs.

solution and tries to iteratively improve it by replacing two edges with two new ones that reduce the tour duration. This algorithm provides a local optimum based on the solution given initially.
To improve the performance of the NSGA-II algorithm we decided to combine it with the 2-Opt algorithm adapted to the MRDS problem, this algorithm is called Hybrid. More precisely, instead of starting with an initial random population, the Hybrid algorithm applies the 2-Opt algorithm to optimize each individual of the initial population. In addition, at each iteration, the children obtained with the crossover operator are mutated with the gene mutation probability and then optimized by applying again the 2-Opt algorithm. As we will see in the next section, the solutions obtained by Hybrid dominate those obtained by NSGA-II with initial solutions randomly chosen.

D. Problem resolution

To solve the MRDS problem, we have implemented both the NSGA-II algorithm and the Hybrid algorithm using java programming language. We evaluated the performance of the NSGA-II algorithm and the Hybrid algorithm using 4 configurations with different numbers of PoIs $\{10, 20, 30, 40\}$ located in a $500m$x$500m$ area. For each configuration, we conducted 30 independent simulation runs using different initial populations randomly selected. Table I shows the simulation parameters used in each configuration, the mutation probability is equal to 0.1.

Number of nodes	Number of robots	Robot capacity	NSGA-II iterations	NSGA-II population size
10	3	10	500	40
20	4	10	500	60
30	5	10	500	80
40	6	10	500	100

TABLE I: Simulation parameters

1) Deployment duration and presence of obstacles: Our goal is to evaluate the solutions provided by both NSGA-II and Hybrid algorithms in terms of the three objectives

considered in the MRDS problem. Each simulation run gives a Pareto front. We then build the final Pareto front of each configuration from the 30 Pareto fronts previously obtained. Furthermore, we quantify the simulation time needed to get these results.

In the real environment, obstacles are always present. They have a big impact on the robot tour and the deployment duration. Thus we study the impact of obstacles. How to compute the deployment duration when one or multiple obstacles exist between two consecutive PoIs of the same robot tour? Figure 4 illustrates a configuration with 40 PoIs depicted in red, the depot depicted in green and three obstacles represented by black rectangles.

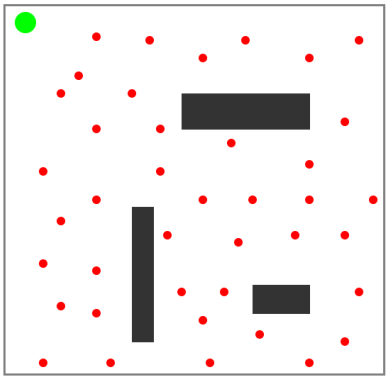


Fig. 4: Configuration with 40 PoIs.

One or several obstacles may exist between two consecutive PoIs in the robot tour. The tour duration increases when obstacles exist since the robot has to bypass these obstacles.

We propose a strategy to bypass the obstacles with the minimum duration. More precisely, we assume that each obstacle is defined by the polygon formed by its vertices. For each obstacle, we define as many intermediate points as the number of obstacle vertices. Then, we select the path that goes through intermediate points until reaching the PoI destination, having the minimum duration. For instance, in Figure 5, the direct path from A to B is made impossible. The intermediate points I_1, I_2 and I_3 are the best combination in terms of duration to go from A to B. The tour duration of this path is computed as the sum of the durations due to any segment composing the path and of course the time needed for angle changes between each two successive segments. Notice that, the list of intermediate points between each two PoIs are precomputed and stored in a file. This information will be used during the simulation runs.

2) Simulation results: When the number of PoIs is small, (e.g. 20 PoIs) both NSGA-II and Hybrid algorithms provide close Pareto fronts. For instance Figure 7a depicts the Pareto fronts obtained when 20 PoIs are deployed in an area without obstacles. However, when obstacles are present, the Pareto front obtained by the Hybrid algorithm

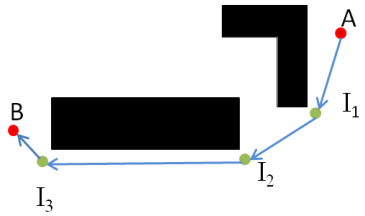
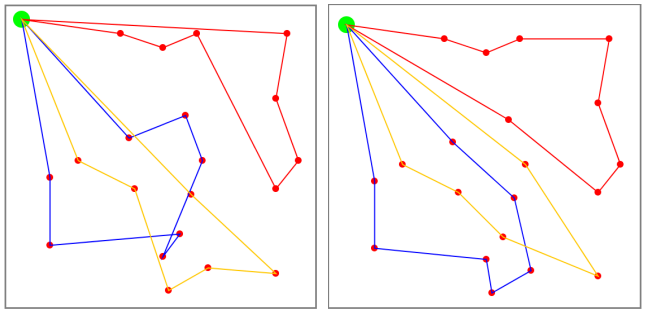


Fig. 5: Intermediate points between sensor node positions A and B

is better in terms of tours duration and tours balancing (i.e. standard deviation) as shown in Figure 7b.

Figures 6a and 6b illustrate the best solution belonging to the Pareto front for 20 PoIs and 3 robots with the smallest maximum tour duration with the NSGA-II and Hybrid algorithms. NSGA-II provides a maximum tour duration of $1416s$ and a standard deviation of 25.32, whereas the Hybrid algorithm gives $1328s$ and a standard deviation of 3.7, respectively. Then, this solution belonging to the Pareto front obtained by Hybrid algorithm dominates the one obtained by the NSGA-II algorithm.



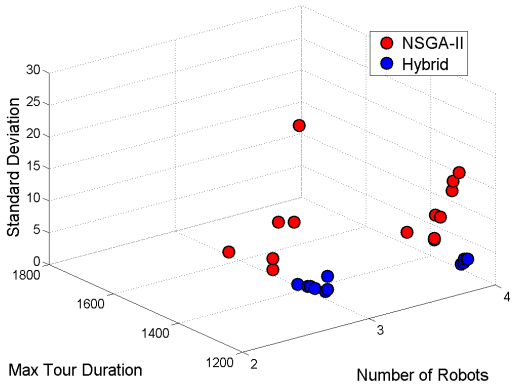
a NSGA-II algorithm. b Hybrid algorithm.

Fig. 6: Tours of 3 robots with 20 PoIs without obstacles.

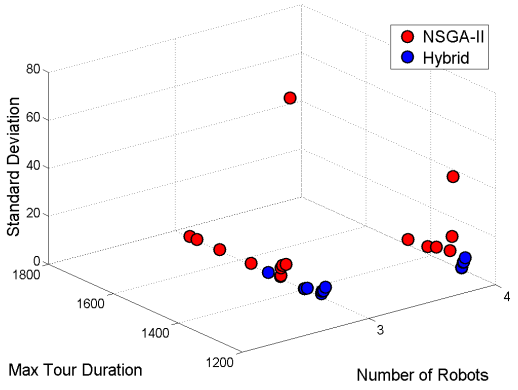
In larger configurations, for instance when 30 or 40 PoIs should be deployed and when obstacles are present or are not, the Pareto fronts obtained by the Hybrid algorithm outperform the ones obtained by NSGA-II in terms of tours duration and tours balancing. This is due to the use of the 2-Opt algorithm that permits to avoid edges crossing in the same tour, leading to smaller tours duration and better balancing between these tours.

To demonstrate the distribution of non-dominated individuals on the objective space for NSGA-II and Hybrid algorithms, we have considered 4 configurations (10, 20, 30 and 40 PoIs) with and without obstacles. Figures 7a, 8a, and 9a depict the Pareto front obtained by gathering all the non-dominated solutions found by each algorithm in the 30 independent runs corresponding to configurations with 20, 30 and 40 PoIs, respectively.

For each possible number of robots, the distribution of non-dominated solutions found by Hybrid is more localized than with NSGA-II, making easier the choice of the suitable solution.



a Without obstacles.

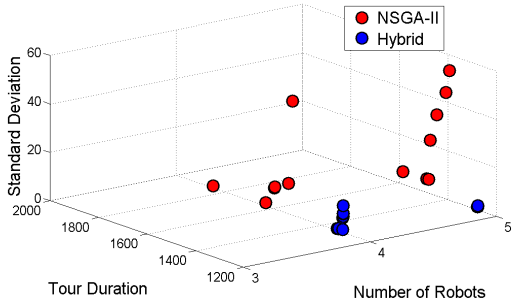


b With obstacles.

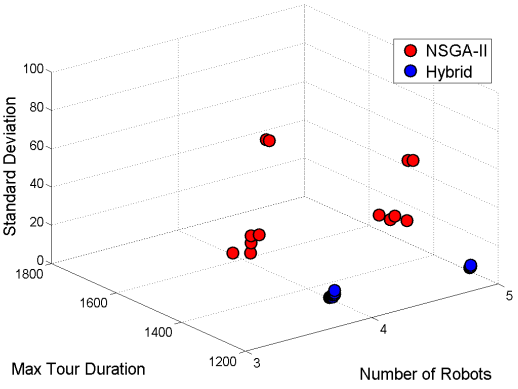
Fig. 7: Pareto front obtained by 20 PoIs.

Figures 7a, 8a, and 9a show the improvement in the quality of the solutions obtained by the Hybrid algorithm with regard to NSGA-II. We can observe that the Pareto front of Hybrid dominates in most cases the Pareto front of NSGA-II. However, this improvement has a cost in simulation time as shown in Table II. All our experiments were conducted using a desktop computer Intel Xeon E5 1620 processor with 8-Core 3.6GHz and 8 Gb of memory.

Let us study the impact of obstacles. Figure 10 depicts the tours of 3 robots visiting 20 PoIs in the presence of obstacles. NSGA-II provides a smallest maximum tour duration of $1443s$ and a standard deviation of 77, whereas Hybrid gives a smallest maximum tour duration of $1334s$ and a standard deviation of 4.8. Clearly, the presence of obstacles leads to larger tour durations and larger simulations times as shown in Table II. As expected, the Pareto front



a Without obstacles.



b With obstacles.

Fig. 8: Pareto front obtained by 30 PoIs.

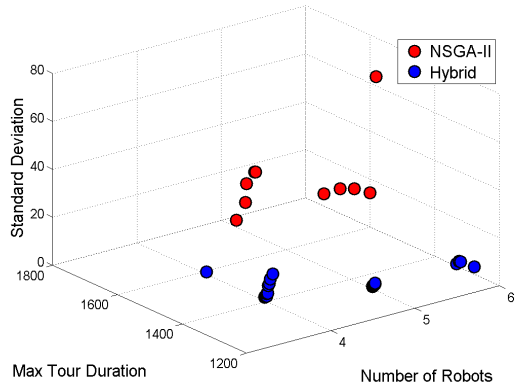
	Without Obstacles				With Obstacles			
	NSGA-II		Hybrid		NSGA-II		Hybrid	
PoI	Ave.	Std	Ave.	Std.	Ave.	Std	Ave.	Std
10	32,07	1,33	75,52	3,43	841,63	71,11	1436,04	84,81
20	133,45	4,56	248	35,39	1502,09	141,38	1520,44	253,14
30	359,17	13,23	557,75	71,01	1709,11	140,2	2296,58	414,62
40	1287	32,38	1027	130,5	2870,81	243,36	3269,3	538,39

TABLE II: The average and the standard deviation of simulation times (in seconds) obtained by NSGA-II and Hybrid algorithms

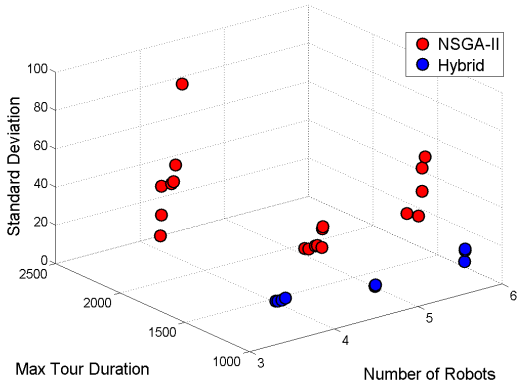
provided by Hybrid dominates in most cases the Pareto front given by NSGA-II.

V. Conclusion

The deployment of static wireless sensor nodes is usually done by mobile robots in charge of placing these sensor nodes at precomputed positions in a given area. Since robots are usually battery-operated and the environment may be hostile, for instance in a post-crisis situation, the duration of the deployment must be minimized. This duration is equal to the longest duration of robot tours. In order to balance these tour durations, we also want to minimize their standard deviation. In addition, the number of robots used

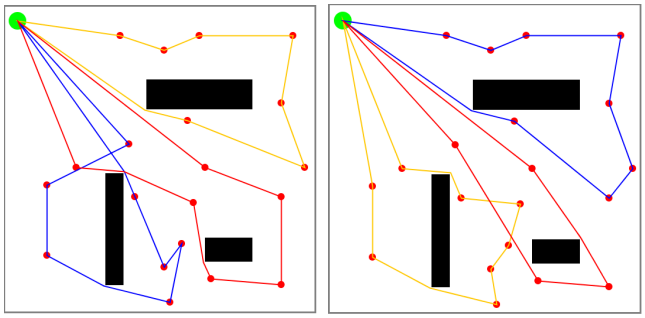


a Without obstacles.



b With obstacles.

Fig. 9: Pareto front obtained by 40 PoIs.



a NSGA-II algorithm. b Hybrid algorithm.

Fig. 10: Tours of 3 robots with 20 PoIs and obstacles.

should be minimized. This multi-objective optimization problem, called MRDS, is formalized. We solve it with NSGA-II and an Hybrid algorithm combining NSGA-II and the 2-Opt heuristic for various configurations (10, 20, 30 and 40 PoIs visited) with and without obstacles.

REFERENCES

[1] C.-Y. Chang, J.-P. Sheu, Y.-C. Chen, and S.-W. Chang, "An obstacle-free and power-efficient deployment algorithm for wireless sensor networks," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 39, no. 4, pp. 795–806, 2009.

[2] C.-Y. Chang, Y.-C. Chen, and H.-R. Chang, "Obstacle-resistant deployment algorithms for wireless sensor networks," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 6, pp. 2925–2941, 2009.

[3] D. Applegate, R. Bixby, V. Chvátal, and W. Cook, "Thetravelingsalesman problem: A computationalstudy, princeton universitypress," ISBN 978-0-691-12993-8, Tech. Rep., 2006.

[4] I. Khoufi, E. Livolant, P. Minet, M. Hadded, and A. Laouiti, "Optimized trajectory of a robot deploying wireless sensor nodes," in *Wireless Days*, 2014.

[5] I. Khoufi, P. Minet, M.-A. Koulali, and M. Erradi, "A game theory-based approach for robots deploying wireless sensor nodes," in *IWCMC*, 2015.

[6] B. M. Baker and M. Ayechew, "A genetic algorithm for the vehicle routing problem," *Computers & Operations Research*, vol. 30, no. 5, pp. 787–800, 2003.

[7] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 2, pp. 182–197, 2002.

[8] M. Hadded, R. Zagrouba, A. Laouiti, P. Muhlethaler, and L. Azouz Saidane, "A multi-objective genetic algorithm-based adaptive weighted clustering protocol in vanet," *IEEE Congress on Evolutionary Computation*, 2015.

[9] N. Kumar and R. K. Karambir, "A comparative analysis of pmx, cx and ox crossover operators for solving traveling salesman problem," *International journal of Latest Research in science and technology*, vol. 1, 2012.

[10] G. A. Croes, "A method for solving traveling-salesman problems," *Operations Research*, vol. 6, no. 6, pp. 791–812, 1958.